

Lab 6

Due: @11:55 PM, Wednesday October 11th

The following assignment is intended to be completed during your assigned lab period. One member of your group must submit the assignment to Gradescope by the posted deadline and indicate your group members when submitting the assignment. **Each group member must be present during the scheduled lab period in order to receive credit.**

Group names and usernames

Pod Member Name	Uniqname
Therron Montgomery	therronm
Yuhan Zhang	zyuhan
Yuzhen chen	yuzhech

For each of the following problems, one person should act as the "scribe" and log the discussions of the group. You should rotate who is the scribe for each problem and indicate in the given space.

For this lab, please also time how long it takes your group to finish the problem.

Problem #	Point Value	Scribe Name	Time Taken (minutes)
1	15		
2	15		
3	20		

Problem 1 (15 points): Single-Cycle vs Multi-Cycle Datapath

Consider the multi-cycle and single-cycle datapath from lecture with the following delays:

Read memory	4 ns
Write memory	4 ns
Read register file	3 ns
Write register file	3.5 ns
ALU	2.5 ns
All other operations	0 ns

- a) What is the minimum clock period for the single-cycle processor? (2 points)

$$4 + 4 + 3 + 3.5 + 2.5 = 8 + 3 + 6 = 17 \text{ ns.}$$

- b) What is the minimum clock period for the multi-cycle processor? (2 points)

$$4 \text{ ns}$$

Consider the following benchmark of 100 instructions. (Ignore the halt instruction.)

Benchmark 1	(100 instructions)
ADD/NOR	15 %
LW	60 %
SW	10 %
BEQ	10 %
NOOP	5 %
JALR	0 %

$$15 \times 4$$

$$60 \times 5$$

$$10 \times 4$$

$$10 \times 4$$

$$5 \times 2$$

- c) What is the execution time of this benchmark on the single-cycle processor? (2 points)

$$100 \times 17 = 1700 \text{ ns.}$$

- d) What is the execution time of this benchmark on the multi-cycle processor? (2 points)

$$15 \times$$

Cycles per Instruction (CPI) is a metric which tells how many cycles are required on average per instruction. As the name suggests it is computed by following formula:

$$CPI = \text{Number of clocks required for benchmark} / \text{Number of instructions executed in the benchmark}$$

- e) What is the CPI of this benchmark on a single-cycle processor? (1 point)

1

- f) What is the CPI of this benchmark on a multi-cycle processor? (2 points)

$$0.15 \times 4 + 0.6 \times 5 + 0.1 \times 4 + 0.1 \times 4 + 0.05 \times 2$$

$$= 0.6 + 3 + 0.4 + 0.4 + 0.1 = 4.5$$

Similarly, we can calculate the average execution time per instruction (TPI) either from the instruction count and execution time, or from the CPI and clock period.

- g) What are the 2 formulas? What are the TPIs for the single and multi-cycle processors on this benchmark? (2 points)

$$TPI = \text{Time} / \# \text{instruction} \quad CPI = \text{clock.p}$$

CPI

- h) Come up with a new benchmark which has lower execution time on the multi-cycle datapath compared to the single-cycle datapath. It can have only the instructions shown in the table below. Fill in the distribution of instructions in this new benchmark. (There are multiple solutions to this problem. You are supposed to come up with any one of them). (2 points)

Benchmark 2	(100 instructions)
ADD/NOR	— % 4
LW	15 % 15
SW	— % 4
BEQ	— % 4

clock: 4

17

4 5

$$17 \div 4 = 4.25$$

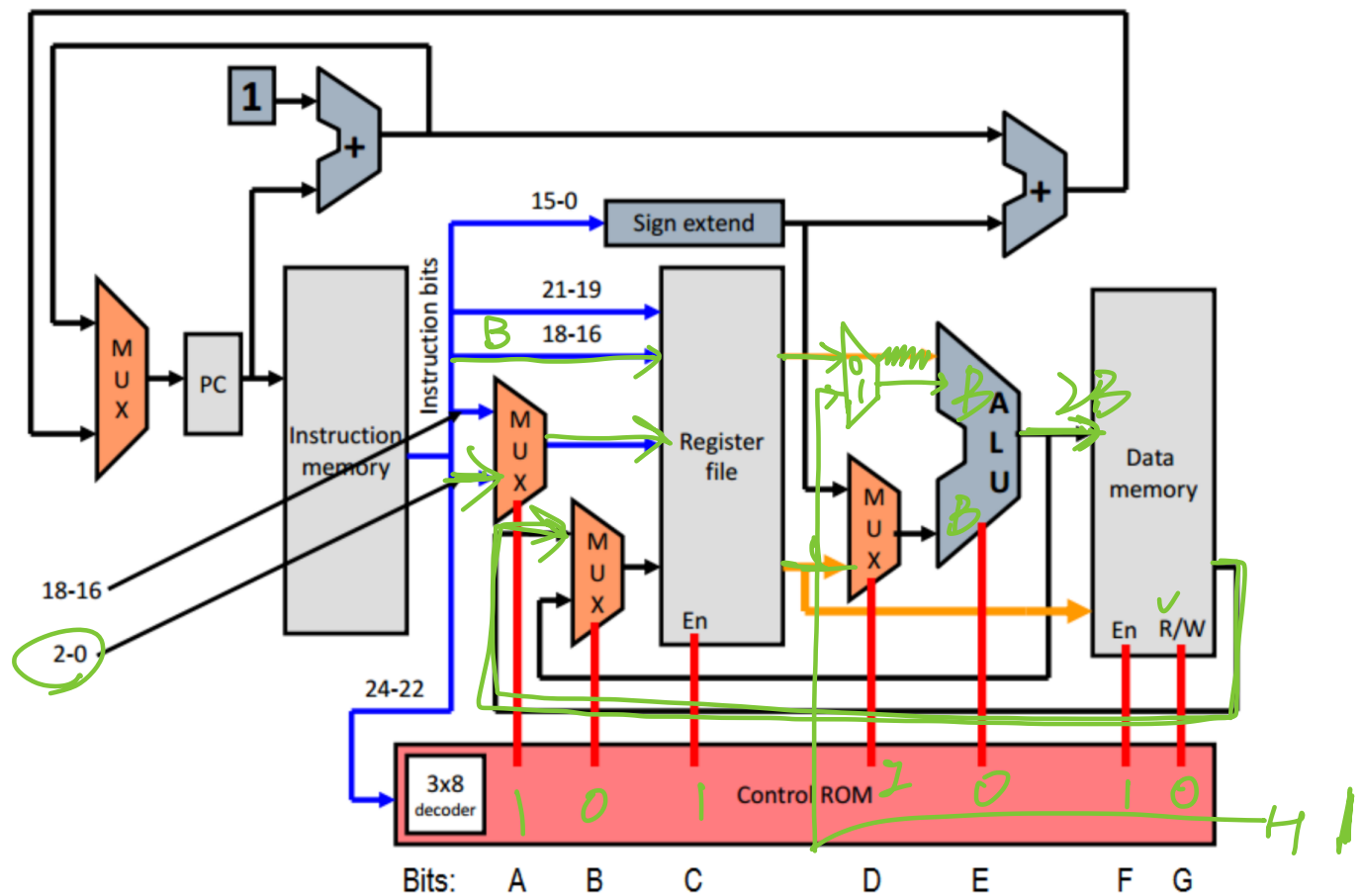
Other	0 %
-------	-----

Problem 2 (15 points). A Single New LC-2K Instruction

Extend the LC2K single-cycle datapath shown below to support a new R-type instruction:

```
[destReg] = memory[ 2* [regB] ]
```

You can add exactly **one** 2:1 mux. We will call this **mux H**. Draw legibly.



A. Specify the input and output connections to the new **mux H** in your modified datapath:

Input 0	Input 1	Output is connected to:
value of A	value of B	Input 0 of ALU.

B. Determine the control signals for the new instruction. (H is new mux H's select signal)

Control Bits:	A	B	C	D	E	F	G	H
	1	0	1	1	0	1	0	1

Datapath reference:

MUX: Setting control to 0 selects the uppermost signal.

ALU: Set control to 0 for add, 1 for nor

Memory: Set control to 0 for read, and 1 for write.

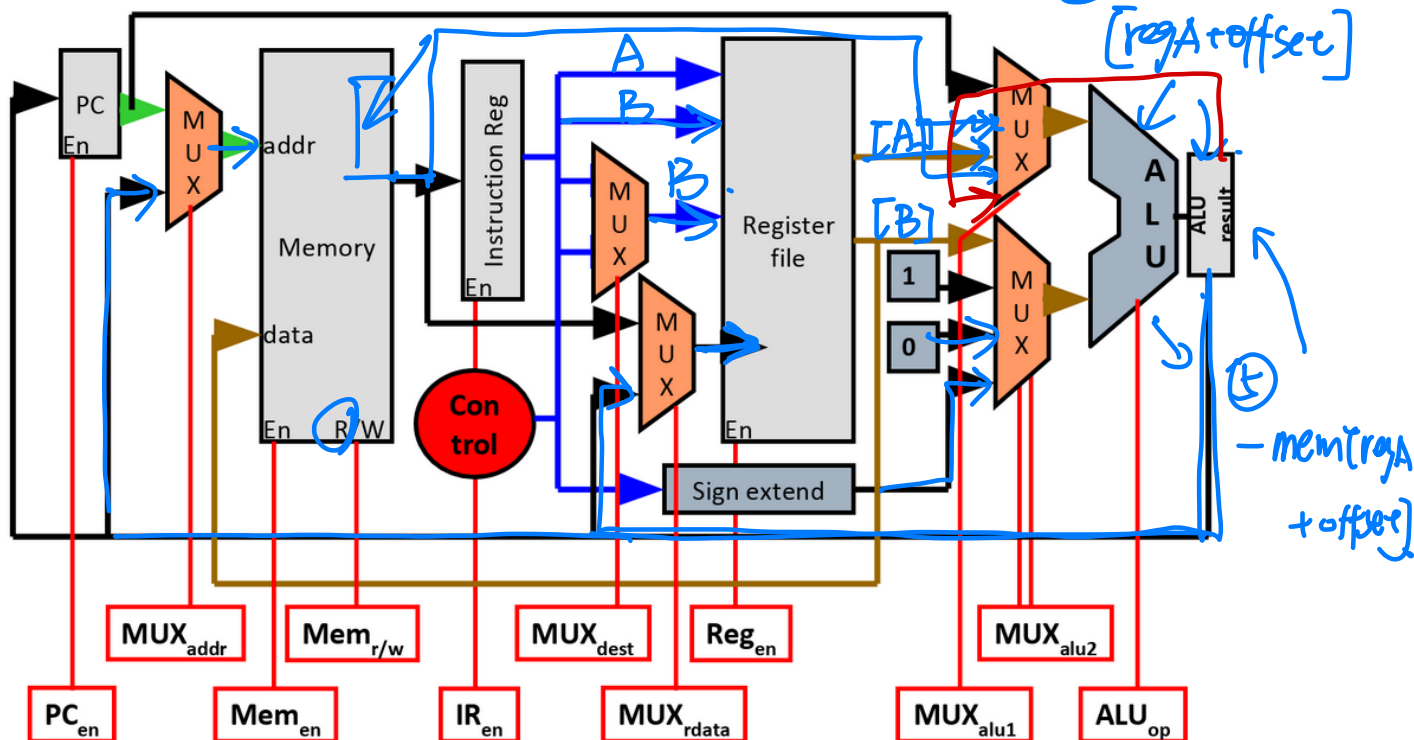
Problem 3 (20 points): Negated Load

A new instruction is proposed to replace `noop` instruction in LC2K. The goal of this instruction is to load the negated 2's complement value of the data in the memory. Following is the format of this new instruction:

`nlw regA regB offset // regB = -mem[regA + offset]`

Read the memory value at address `regA + offset`, and store the negated 2's complement version in regB.

Example: `nlw 0 2 4 // reg2 = -mem[reg0 + 4]`









To support this new instruction, The above multicyle datapath needs to be modified.

PC (en)	MUX (addr)	MEM (en)	MEM (r/w)	IR (en)	MUX (dest)	MUX (rdata)	Reg (write en)	MUX (alu1)	MUX (alu2)	ALU (op)
0	0	1	0	1	X	X	0	0	01	0
1	X	0	X	0	X	X	0	X	XX	X
0	X	0	X	0	X	X	0	01	11	0
0	1	1	0	0	X	X	0	XX	XX	X
0	X	0	X	0	X	X	0	10	10	1
0	X	0	X	0	X	X	0	11	01	0
0	X	0	X	0	0	1	1	XX	XX	X

1
2
3
4
5
6
7.

a) To support **n1w** instruction, you can only modify the MUXes (and add any necessary connections to them) **in the table below**. Note that none of the MUXes should have more than four inputs to them. Describe your changes to datapath below. Note **n1w** takes **7** cycles to complete in the new datapath, and decoding is done the same way as **1w** instruction. (5 points)
































MUX name	Modified (Yes/ No)	Connections added/removed
MUX_addr		\
MUX_dest		\
MUX_rdata		\
MUX_alu1		
MUX_alu2		\

b) Write the control signals for all the cycles needed for the “**n1w**” instruction on the modified datapath after your connections have been added. (15 points)

You can assume the following things:

- The top mux input is selected when all mux control bits are 0
- All connections added to any mux are added at the bottom of the mux
- The select bits for extended muxes may need to be increased from the original mux
- Mem (r/w): 0 for read, 1 for write
- ALU (op): 0 for add and 1 for nor

Hint. If you've added more inputs to the MUX, make sure the number of control bits for the modified mux is correct

PC (en)	MUX (addr)	MEM (en)	MEM (r/w)	IR (en)	MUX (dest)	MUX (rdata)	Reg (write en)	MUX (alu1)	MUX (alu2)	ALU (op)
0	0	1	0	1	X	X	0	0	01	0
1	X	0	X	0	X	X	0	X	XX	X
0	X	0	X	0						
0				0	X	X	0			
0				0	X		0			
0	X	0	X	0						
0	X	0	X	0						

For time in lab, boxes in blue have been filled in for you.